Attorney Docket No. 7744-0061

09887528.052501

UNITED STATES PATENT APPLICATION

OF

Michael A. EKHAUS

Robert DRISKILL

Filip MULIER

FOR

METHOD AND SYSTEM FOR HIGH PERFORMANCE MODEL-BASED PERSONALIZATION

RELATED APPLICATIONS

This application claims benefit of United States Provisional Application No. 60/213,528 entitled "High performance model-based personalization," by Ekhaus et al. filed on June 23, 2000, the entire contents of which are hereby incorporated, and from which priority is claimed.

09887528.062501

I. FIELD OF INVENTION

The present invention relates to the field of personalization systems. More particularly, the present invention relates to a method and system for generating client preference recommendations in a high performance computing regime.

II. BACKGROUND OF THE INVENTION

In a conventional transaction in which a client selects a good or service, the client generally has a set of preferences associated with a similar or dissimilar set of goods or services. In a mathematical sense, one can make a one-to-one correspondence between the set of preferences and the set of goods or services. For example, given the ordered set of goods or services:

{*book* "***A1***", *film* "***A2***", *restaurant* "***A3***", … , *music* "***A200***"}

where "***A1-A200***" denote, for example specific products or services, and given client "***U***", the ordered preferences may be expressed as for example Boolean quantities:

U*'s preferences*:

{*book* "***A1***": yes, *film* "***A2***": not yes, *restaurant* "***A3***": yes, … , *music* "***A200***": yes}

This may be expressed in the shorthand form, using "1" to denote "yes", "0" to denote "not yes", and "***U***" to denote "***U's preferences***" as a row vector containing 200 entries:

$\boldsymbol{U} = \{1, 0, 1, \ldots, 1\}$

Suppose that there is an additional item of interest: *book* "***A201***," of which ***U***'s Boolean preference is unknown: "0". Denoting *book* "***A201***" in the ordered set as one increment to the right, one may express this as:

$\boldsymbol{U} = \{1, 0, 1, \ldots, 1, 0\}$

Suppose, further, that the following preferences for other clients, ***Z1-Z1000*** are known and encompass *book "A201"*:

Z1 = {0, 1, 0, … , 1, 1}

Z2 = {1, 1, 1, … , 0, 1}

Z3 = {1, 1, 0, … , 1, 0}

…

Z1000 = {0, 0, 1, … , 1, 1}

Personalization systems are generally designed in order to provide a robust recommendation for client ***U*** regarding item ***A201,*** based upon known preferences of client ***U*** as well as known preferences of other clients ***Z1-Z1000.*** The use of conventional personalization systems or recommendation systems in E-commerce is described in "Recommender Systems in E-Commerce," *Proceedings of the ACM Conference on Electronic Commerce*, Nov. 3-5, 1999, by Schafer *et al.*

One skilled in the art will appreciate that the utility of a recommendation system is driven by the method used for determining the amount of correlation that exists between the votes for two or more items, or by the amount of correlation that exists between the votes of two or more clients. There are a number of ways of determining correlation, for example, as discussed in "An Algorithmic Framework for Performing Collaborative Filtering," *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, Aug. 1999, by Herlocker *et al.* Such methods include, for example the computation of Pearson correlations, as used in the GROUPLENS system, the calculation of Spearman rank correlation coefficients, or a least-squares comparison.

A basic problem with conventional recommendation systems, however, is directly related to the issue of combinatorial explosion. The volume of data collected from clients engaged in E-commerce is outpacing the conventionally applied computational ability to rapidly process such preferences and generate accurate recommendations. Although the introductory examples articulated in this document represent relatively trivial matrices (*i.e.* 201 × 201 matrices, or 1,000 × 1,000 matrices), in actual practice one must be able to work with matrices of the order of 1,000,000 × 1,000,000 and higher. In light of the foregoing, it remains desirable to introduce a system and method that can accurately process large ratings-matrices in a rapid fashion so as to generate accurate recommendations.

Another concern with conventional systems is related to the desire to preserve client data privacy. With such a large amount of data being processed for a given client, it is desirable for a system and method that will not allow one to reconstruct the original data set from the disclosed portion of the recommendation model.

III. SUMMARY OF THE INVENTION

Accordingly, in a first embodiment of the present invention, a method of providing a recommendation to a user comprises: providing a sparse ratings matrix, forming a plurality of data structures representing the sparse ratings matrix, forming a runtime recommendation model from the plurality of data structures, determining a recommendation from the runtime recommendation model in response to a request from a user, and providing the recommendation to the user.

In a second embodiment of the present invention, a method of providing a recommendation to a user comprises: providing a sparse ratings matrix, providing an update ratings data structure, forming a plurality of data structures representing the sparse ratings matrix, forming a runtime recommendation model from the plurality of data structures and the update ratings data structure, determining a recommendation from the runtime recommendation model in response to a request from a user, and providing the recommendation to the user.

In a third embodiment of the present invention, a method of providing a recommendation to a user comprises: providing a sparse ratings matrix, forming a plurality of data structures representing the sparse ratings matrix, forming a first recommendation model from said plurality of data structures, perturbing the first recommendation model to generate a runtime recommendation model, determining a recommendation from the runtime recommendation model in response to a request from a user, providing the recommendation to the user.

In a fourth embodiment of the present invention, a method of providing a recommendation to a user comprises: providing a sparse ratings matrix, forming a plurality of data structures representing the sparse ratings matrix, forming a first recommendation model from the plurality of data structures, truncating the first recommendation model to generate a runtime recommendation model, determining a recommendation from the runtime recommendation model in response to a request from a user, and providing the recommendation to the user.

In a fifth embodiment of the present invention, a method of providing a recommendation to a user comprises: providing a first ratings matrix, providing a second ratings matrix, forming a runtime recommendation model from the cross-set co-occurrences of the first ratings matrix and the second ratings matrix, determining a recommendation from the runtime recommendation model in response to a request from a user, and providing the recommendation to the user.

Further still, in a sixth embodiment of the present invention, a method of providing a recommendation to a user comprises determining a recommendation from a recommendation model using a multiplicity voting scheme, which may be personalized or may be anonymous.

Additional features and advantages of the invention will be set forth in the description that follows, and in part will be apparent from the description, or may be learned by practice of the invention. The objectives and other advantages of the invention will be realized and attained by the process and apparatus particularly pointed out in the written description and claims herein as well as the appended drawings.

IV. BRIEF DESCRIPTION OF DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an implementation of the invention and, together with the description, serve to explain the advantages and principles of the invention. In the drawings,

FIG. 1 depicts a recommendation scheme of the prior art in which a base model is not modified before generating a preference recommendation through an on-line or runtime model;

FIG. 2 depicts a recommendation scheme consistent with the present invention in which additional data allows the construction of a perturbed on-line model and generates a modified runtime recommendation model;

FIG. 3 schematically indicates exemplary relationships between various processes of the present invention and various helper functions in preferred embodiments;

FIG. 4 depicts a system configuration consistent with the present invention in which a runtime recommendation system cooperates with an off-line recommendation system;

FIG. 5 is a schematic depiction of a method consistent with a first embodiment of the present invention;

FIG. 6 depicts a conventional a node structure of the prior art in a distributed computing process;

FIG. 7 is a schematic depiction of a method consistent with a second embodiment of the present invention;

FIG. 8 is a schematic depiction of alternative methods consistent with a third or fourth embodiment of the present invention;

FIG. 9 is a schematic depiction of a method consistent with a fifth embodiment of the present invention; and

FIG. 10 depicts an example from the prior art of a transformation to compressed row format.

V. DETAILED DESCRIPTION

Reference will now be made in detail to an implementation consistent with the present invention as illustrated in the accompanying drawings. Whenever possible, the same reference number will be used throughout the drawings and the following description to refer to the same or like parts.

V.A. Term Definitions

As used herein, the symbol $\Re$ indicates a set. An array of the form

$$R = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \cdots & r_{1,m} \\ r_{2,1} & r_{2,2} & r_{2,3} & \cdots & r_{2,m} \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ r_{n,1} & r_{n,2} & r_{n,3} & \cdots & r_{n,m} \end{pmatrix}$$

with entries with values $r_{i,j} \in \Re$ is called an $n \times m$ matrix over $\Re$, with n rows and m columns. One skilled in the art should appreciate that the matrices in question are such that n and m are finite but not bounded. For example, new rows and columns are often added. This implies that calculations are preferably performed over infinite matrices with the property that all entries are zero except for those entries in a finite number of rows and columns.

As used herein, a row from a matrix, for example $\boldsymbol{R}$, is a $1 \times m$ matrix. If i indicates the row in question, then

$$(r_{i,1}\ r_{i,2}\ r_{i,3} \cdots r_{i,m})$$

is the ith row of $\boldsymbol{R}$. Similarly, as used herein, a column from a matrix is an $m \times 1$ matrix. If j indicates the column in question, then

$$\begin{pmatrix} r_{1,j} \\ r_{2,j} \\ r_{3,j} \\ \cdot \\ \cdot \\ \cdot \\ r_{m,j} \end{pmatrix}$$

is the *j*th column of ***R***.

As used herein, a *vector* is a 1 × *m* or a *n* × 1 matrix. The above statement regarding unbounded matrices applies to vectors as well. Accordingly, one skilled in the art should appreciate that definitions of operations are depicted over finite ranges for convenience only.

As used herein, *rating* indicates an entry of a "rating matrix," defined below. The presence of an entry is an indication that a relationship exists between a given *client* and a given *item*.

As used herein, a *ratings matrix* is a collection of numerical values indicating a relationship between a plurality of clients and a plurality of items. In general, and as indicated earlier, one may denote this as:

$$\boldsymbol{R} = \boldsymbol{R}_{u,i} = \begin{cases} 1 : \text{if client } u \text{ votes favorably for item } i \\ 0 : \text{otherwise} \end{cases}$$

where $u \in U$, the set of all clients, and $i \in I$, the set of all items. One skilled in the art should appreciate that "votes favorably" as used above may correspond to a variety of acts. For example, a favorable vote may correspond to client *u* purchasing item *i*, or it may correspond to client *u* literally expressing a favorable interest in item *i*. Again, item *i* itself is not limited to goods but may also correspond to services.

As used herein, the notation $A_{i,*}$ will denote the *i*th row of matrix A and $A_{*,j}$ will denote the *j*th column of A. Further still, it is useful to speak of vectors as if they were sets and vice-versa. One skilled in the art should be able to discern which is being referred to by the context of the operations performed. If one considers the set

$$\boldsymbol{R}_{*,i} \Leftrightarrow \{ u \in U \mid \boldsymbol{R}_{u,i} = 1 \}$$

given any two items *i* and *j* ∈ *I*, then the set of clients having voted favorably for both items is given by:

$$\boldsymbol{R}_{*,i} \cap \boldsymbol{R}_{*,j} = \{u \in U \mid \boldsymbol{R}_{u,i} = 1\} \cap \{u \in U \mid \boldsymbol{R}_{u,j} = 1\}$$

$$= \{u \in U \mid \boldsymbol{R}_{u,i} = 1 , \boldsymbol{R}_{u,j} = 1\}$$

$$= \{u \in U \mid \boldsymbol{R}_{u,i} \boldsymbol{R}_{u,j} = 1\}$$

Furthermore, if the cardinality of the set is taken, then the following is derived:

$$\#\left(\boldsymbol{R}_{*,i} \cap \boldsymbol{R}_{*,j}\right) = \sum_{u \in U} \boldsymbol{R}_{u,i} \boldsymbol{R}_{u,j} = \sum_{u \in U} \boldsymbol{R}^{t}_{i,u} \boldsymbol{R}_{u,j} = \left(\boldsymbol{R}^{t} \boldsymbol{R}\right)_{i,j}$$

The above relationship indicates that the dot product of the two columns from the ratings matrix is a sum over the number of co-rates between two items. Performing this for all possible pairs yields an *item-item* matrix of co-rates.

As used herein, an *item-item model* may be constructed by computing the matrix ${}_{(I\text{-}I)}\boldsymbol{M} = \boldsymbol{R}^{t} \boldsymbol{R}$ where the superscript "*t*" indicates a transposed matrix, and the pre-subscript "*(I-I)*" on $\boldsymbol{M}$ indicates an *item-item model.* The *item-item* model indicates the correlation between two items for which preference ratings are known. The diagonal portion of $_{(I\text{-}I)}\boldsymbol{M}$, for example the entry at row *i* and column *i*, corresponds to the total number of votes for item *i*. Furthermore, the number of clients having co-rated any item-item pair is given by the respective entries from the matrix $\boldsymbol{R}^t\boldsymbol{R}$.

Further still, given any two clients, the number of co-rated items between them is given by the respective entry of $\boldsymbol{R}\boldsymbol{R}^t$. Both symmetric forms are of interest to the types of problems that will be discussed herein.

Accordingly, and as used herein, a *client-client* model may be constructed by computing $_{(c\text{-}c)}\boldsymbol{M} = \boldsymbol{R}\,\boldsymbol{R}^t$, where the pre-subscript "*(c-c)*" indicates "*client-client.*" As before, the diagonal entries of the above matrix indicate how many favorable votes the corresponding client made.

As used herein, the *item-item model* $_{(I\text{-}I)}\boldsymbol{M}$ and the *client-client* model $_{(c\text{-}c)}\boldsymbol{M}$ will be denoted in general as ***M***.

One skilled in the art should appreciate that, given a ratings matrix ***R***, then $\boldsymbol{R}^t\boldsymbol{R}$ as well as $\boldsymbol{R}\,\boldsymbol{R}^t$ are symmetric as previously noted. For any given row of $\boldsymbol{R}^t\boldsymbol{R}$ or $\boldsymbol{R}\,\boldsymbol{R}^t$, the diagonal entry is the largest entry in the row. This is made apparent by considering that for each *i*, one has for all *j* that

$$\boldsymbol{R}_{*,i} \cap \boldsymbol{R}_{*,j} \subseteq \boldsymbol{R}_{*,i}$$

In addition, given any row *i*, the value of the diagonal term is the number of non-zero entries in the *i*th column of ***R***. Therefore, given any column index, *i*, of ***R*** the *i*th row (or column) of $\boldsymbol{R}^t\boldsymbol{R}$ or $\boldsymbol{R}\,\boldsymbol{R}^t$ induces a relative scaling on all column indices of ***R***. One may order the column indices according to this scaling, if it is decided how to order between indices that have the same relative ranking. One suitable manner is to decide uniformly between equivalently ranked indices of a row of $\boldsymbol{R}^t\boldsymbol{R}$ or $\boldsymbol{R}\,\boldsymbol{R}^t$.

As used herein, *unary data* indicates a ratings data in which there are only two types of information: *positive* and *no information*. Such data sources are usually encoded with rating values of either zero or one. It is customary to let zero express *no information* since such use produces a sparse data set.

As used herein, *interest data* indicates ratings data in which there is a scaling to the positive interest of a rating. One skilled in the art should appreciate that the range of values is bounded such that each value is finite.

As used herein, *likert data* indicates ratings data in which there is a scaling that includes both positive interest and a degree of possible dislike.

As used herein, *co-rate* indicates either a *co-rate of clients*, or a *co-rate of items*. These two senses are analogous to each other. One skilled in the art should appreciate that two items are said to co-rate each other if and only if there exists a client that has rated both of these items. Therefore, it is permissible to have an item co-rate itself. Further still, two clients are said to co-rate each other if and only if there exists an item that both clients have rated.

V.B. Functional Definitions

The general process of generating a recommendation from a recommendation model consistent with embodiments of the present invention is discussed in this section. As used herein, the function **Index**(*) operating on a row of a matrix (a vector) sets the rows' co-rate with itself to zero. For example, given the row:

$$\boldsymbol{M}_{i=12,*} = \{0,0,1,0,4,2,0,0,0,3,0,6,1,0,0,0\}$$

yields:

$$\mathbf{Index}(\boldsymbol{M}_{i=12,*}) = \{0,0,1,0,4,2,0,0,0,3,0,0,1,0,0,0\}$$

As used herein, the top-*k* co-rate for a row is denoted by $\mathbf{Index}_k$. For example, if $k = 3$

$$\mathbf{Index}_{k=3}(\boldsymbol{M}_{i=12,*}) = \{0,0,0,0,4,2,0,0,0,3,0,0,0,0,0,0\}$$

Notice that if one is interested in the top four co-rates, then a problem of breaking up ties would arise. This is a problem of a local degeneracy within a row. One may break this local degeneracy in a number of ways. For example, the global popularity of the items in question yield several approaches, two approaches of which are to select the most globally popular or the least globally popular. One skilled in the art should understand that when a top-*k* row vector is discussed, a method for breaking ties (or breaking such degeneracies) is implied.

As used herein, the operator that returns the top-*k* values is denoted by $\mathbf{TOP}_k$. For $k = 3$, one has

$$\mathbf{TOP}_{k=3}(\boldsymbol{M}_{i=12,*}) = \{0,0,0,0,4,0,0,0,0,3,0,6,0,0,0,0\}$$

In general, the collection of methods covered by the ***M***-model approach maps a row of ***M*** to a vector using a function of $\mathbf{Index}_k(\boldsymbol{M}_{i,*})$ and some statistics of ***M***. This is usually done to scale the ranking induced by the co-rate matrix. This will be discussed in more detail later, but the most basic of operations is to set all the non-zero co-rates to 1. As used herein, this operation is denoted by **Unary** and in this example

$$\mathbf{Unary}(\mathbf{Index}_{k=3}(\boldsymbol{M}_{i=12,*})) = \{0,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0\}$$

Because of its common use in the on-line recommendation, the above operation will be denoted herein in shortened form herein by $\mathbf{Unary}_k(*)$. If any of these operators act on a matrix, it is defined herein to return a matrix in which the operator acts on each *row* of the input matrix.

As used herein, the *diagonal* operator, ***D***, is an overloaded operator in the following sense: if ***D*** operates on square matrix then the return is a vector whose terms are from the diagonal of the respective matrix; alternatively if ***D*** operates on a vector then it returns a diagonal matrix whose non-zero diagonal entries correspond to the respective vector. For example, given a vector $\vec{v}$, ***D*** is defined herein by

$$\boldsymbol{D}(\vec{v}) = \boldsymbol{D}(\vec{v})_{i,j} = \begin{cases} (\vec{1}^{t} \bullet \vec{v})_{i,i} & : \text{if } i = j \\ 0 & : \text{otherwise} \end{cases}$$

where $\vec{1} = \{1,1,1,\ldots\}$ is a row vector of all 1's. Furthermore, given a square matrix A, then

$$\boldsymbol{D}(A) = \boldsymbol{D}(A)_{1,i} = A_{i,i}$$

As used herein, a *multiplicity voting recommendation* scheme returning a maximum of k' elements from $\boldsymbol{S}$ and k-neighbors is given by

$$\mathbf{Unary\_Multi\_Vote}_{k'}(\boldsymbol{S}, k) = \mathbf{TOP}_{k'}\left(\sum_{i \in \boldsymbol{S}}{}' \mathbf{Unary}_k(\boldsymbol{M}_{i,*}) \right)$$

where the primed summation indicates a summation of the unique entries of $\boldsymbol{S}$ and where $\boldsymbol{S} = \{X_1, X_2, \ldots, X_p\} \in \boldsymbol{X}^p$. As defined herein, the variable X_m may represent an *item* when using the *item-item* model ${}_{(I\text{-}I)}\boldsymbol{M}_{i,*}$, or it may represent a *client* when using the *client-client* model ${}_{(c\text{-}c)}\boldsymbol{M}_{i,*}$.

As used herein, a *non-unary* version of this scheme may be expressed as

$$\textbf{Multi\_Vote}_{k'}(\boldsymbol{S}, k) = \textbf{TOP}_{k'}\left(\sum_{i \in \boldsymbol{S}}' \textbf{Index}_{k}(\boldsymbol{M}_{i,*}) \right)$$

Suppose that ${}_{(I\text{-}I)}\boldsymbol{M} = \boldsymbol{R}^{t}\boldsymbol{R}$ is a base model from which one constructs an on-line model, ${}_{(I\text{-}I)}\boldsymbol{M}^{r(k)} = \textbf{Unary}_{k}({}_{(I\text{-}I)}\boldsymbol{M})$. The derived model is computed so that

$\textbf{Unary\_Multi\_Vote}_{k'}(*, k)$ may be computed more efficiently as

$$\textbf{Unary\_Multi\_Vote}_{k'}(\boldsymbol{S}, k) = \textbf{TOP}_{k'}\left(\sum_{X \in \boldsymbol{S}}' (\boldsymbol{M}_{i,*}^{r(k)}) \right)$$

where *r(k)* indicates the runtime model's dependence on the parameter *k*. This equation represents the unperturbed recommendation system using anonymous recommendations. For personalized recommendation one may use

$\textbf{Unary\_Multi\_Vote}_{k'}(\boldsymbol{R}_{u,*}, k)$ where $u \in \boldsymbol{U}$

Suppose that there are multiple sources of ratings data. These different data sets may represent transactions other than purchases. For example, one set might be client purchases and another might be demographic data for these clients. As another example, the data may represent different divisions from a given company, different companies data, purchases in specified categories, etc. Therefore, suppose that one has a sequence of ratings matrices that represent different *dimensions*

$\boldsymbol{R}^{(1)}, \boldsymbol{R}^{(2)}, \ldots, \boldsymbol{R}^{(p)}$

and that these matrices represent data for a common set of clients. Specifically, suppose that the *i*th row of each matrix represents the same client. As used herein, a ratings matrix of augmented matrices comprises

$$\boldsymbol{R}^{(1)},\boldsymbol{R}^{(2)},\ldots,\boldsymbol{R}^{(p)}$$

$$\mathcal{R}=\{\boldsymbol{R}^{(1)},\boldsymbol{R}^{(2)},\ldots,\boldsymbol{R}^{(p)}\}$$

Computing the matrix of co-rates between different dimensions as $\mathcal{R}^{t}\mathcal{R}$ gives a block matrix whose blocks are given by

$$(\mathcal{R}^{t}\mathcal{R})_{\text{block } i,j}=(\boldsymbol{R}^{(i)})^{t}\boldsymbol{R}^{(j)}$$

Therefore, if one wants to know the top-*k* co-rated members from dimension *j* for dimension *i*, one determines

$$\mathbf{Unary}_{k}\,(\boldsymbol{R}^{(i)})^{t}\boldsymbol{R}^{(j)}$$

The case $i = j$ is the case where there is one source of ratings data $\boldsymbol{R}$ described above. In this case

$$\mathbf{Unary}_{k}(\boldsymbol{R}^{t}\boldsymbol{R}) = M^{r(k)}$$

As used herein, the runtime model of co-rates between dimensions *i* and *j* is

$$M(i,j)^{r(k)} = \mathbf{Unary}_{k}((\boldsymbol{R}^{(i)})^{t}\boldsymbol{R}^{(j)})$$

In this situation, if a recommendation is from dimension *i*, then this recommendation, as used herein, is referred to as an *i*-recommendation. Furthermore, if dimensions *i* and *j* are both being considered for recommendations then it is referred to as $\{i,j\}$-recommendation, etc. Furthermore, *i*-ratings, as used herein, refer to the use of ratings data from dimension *i* as input. As an example, suppose one wants to make *j*-recommendations from *i*-ratings for client *u*. Extending the previously defined approach, this is given by

$$\textbf{Unary\_Multi\_Vote}(i,j)_{k'}(\boldsymbol{R}_{u,*},k) = \textbf{TOP}_{k'}\Big(\sum_{z \in \boldsymbol{R}_{u,*}} (M(i,j)^{r(k)}{}_{z,*})))$$

VI. OVERVIEW OF THE PRESENT INVENTION

Data may be represented in a variety of forms and may correspond to a variety of items of interest. One of the objects of a recommendation model, however, is to draw out correlations in the data between items to aid profitability. The present invention, in a general sense, uses data to build a recommendation model that in order to provide personalized recommendations. In particular, the present invention, in one embodiment, involves constructing multiple recommendation models and from the collection of models, solving a particular client's problem. For example, recommendation models may be characterized as on-line (or runtime) and/or off-line. Further, the on-line recommendation model may be constructed from an off-line model. In certain instances, the off-line model may be better suited for batch processing of recommendations, because performance is less of an issue than that of the on-line equivalent. The usage scenario described below is an example of this situation.

Furthermore, even the off-line model may be only a portion of a larger model. One of the aspects of this approach is that a taxonomy of recommendations may be constructed using various models produced from data sources in an off-line manner. To be more precise, the various models are produced in a manner independent of their use in making personalized recommendations. In this manner, the runtime models are constructed as though in a memory cache in which data has been ordered and as much pre-computation as possible has occurred in anticipation of the final on-line calculations required at runtime.

In summary, it is beneficial to have a methodology that allows one to derive models from previously constructed models, and from which there can be incremental updates, thereby providing current and accurate knowledge of the data represented. These models, in turn, may be used in either an off-line or on-line fashion to construct recommendations. One skilled in the art should appreciate that one of the benefits of the present invention is the development of a consistent interpretation of correlated data as it relates to doing business across multiple interactions with a client. Additional data derived from such interactions may then be fed back into existing data, thereby allowing the process of model creation to incrementally update the collection of models to from the most up-to-date and accurate knowledge for both off-line and on-line (runtime) processing. The accuracy of such a collection of models is a measure of at least two aspirations: (i) firstly, the ability to correctly represent all data sources contributing to the models; and (ii) secondly, the ability to correctly represent the current (or runtime) intentions of the recommender (for example, the marketer or the reseller).

VI.A. Usage scenario

For exemplary purposes only, suppose that a widget reseller has an Internet site or a call center at which clients may buy a plurality of widgets. Furthermore, the following information is considered known: (1) historical order data; (2) categorization of the widgets; and (3) profit margins for each widget

Suppose that the widget reseller has a recommendation model, which provides client recommendations with respect to the plurality of widgets offered for sale. At some point in time, the widget reseller determines that a first widget is overstocked. Thus, the widget reseller needs to sell the first widget. Accordingly, and based on the updated information regarding the first widget, the widget reseller would like to: (i) determine likely previous clients to target, using, for example, direct mailing, e-mail, or the telephone; and (ii) if a client comes to the Internet site of the widget reseller and the reseller determines that the client is a strong candidate to buy a first widget, then widget reseller would like to detect this event and recommend a first widget to the client.

Accordingly, various embodiments of the present invention perform these two actions from a common framework while: (i) using the existing recommendation model for making recommendations as a base and deriving a modified model that represents the need to sell first widget; and (ii) changing the current working model for an on-line recommendation to reflect this need. That is to say, if the widget reseller would typically recommend a second widget under the base model, with the caveat that the second widget is part of a recommendation category in the base model that also contains the first widget, then the widget reseller may prefer to recommend the first widget in the modified recommendation model in place of the recommendation for second widget under the base model.

Furthermore, suppose that the profit margin for the first widget is median among the range of profit margins for other widgets that may be likewise recommended under a base model. In such an instance, and under a modified recommendation model, the widget reseller may want to replace a recommendation for all those widgets with a profit margin less than first widget with a recommendation for the first widget. Further still, the widget reseller may want to recommend the first widget in place of recommendations for widgets whose profit margin is higher than that of first widget, under a modified recommendation model. One skilled in the art should appreciate that this may be desirable in the case where one does not want to ignore high profit margins at the expense of removing an over stocked, lower profit margin widget.

FIG. 1 depicts a recommendation scheme of the prior art where a recommendation model is not modified to reflect additional data, such as where a given item, say a first widget, should preferably be recommended more often or less often.

FIG. 2 depicts a modified situation consistent with the present invention where the dashed arrow schematically indicates the influence of additional data 60 as a result of introducing perturbed on-line model 210. Such a modification of base model 110, for example, uses additional data 60 to generate a marketing campaign for an item, say first widget, and perturbed on-line model 210 derived from the base model 110. As discussed in more detail below, one skilled in the art should appreciate that the perturbation may be undone at a time when it is decided to revert back to the base model. For example, at some time in the future (after the overstocked first widgets have been sold) the base model may be accepted as accurately reflecting ordinary, unperturbed buying patterns.

This above scenario is exemplary only, and is intended to illustrate one use of the described invention. Additionally, in the discussion that follows, one skilled in the art should appreciate that the examples disclosed herein are expressed as dense matrices and/or dense vectors for purposes of readability. However, the methods and systems disclosed herein relate generally to sparse matrices and/or sparse vectors.

FIG. 3 indicates the relationship between the various processes consistent with the present invention. In particular, FIG. 3 is an exemplary schematic of the flow of data in an embodiment of the present invention, comprising the processes of: (1) preprocessing data 55; (2) scheduling 40; (3) loading a model 405; (4) adding rating(s) 65; (5) initializing 150; (6) updating 140; (7) perturbing model 35; (8) making a personalized recommendation 310; and (9) making an anonymous recommendation 320. FIG. 3 further illustrates the partitioning of off-line processing region 105 and runtime processing region 400.

Each of the above processes is described in more detail below. Items (1) through (4) above, in a preferred embodiment of the present invention, pertain to processing that assists the overall function of model creation. Items (5) through (9), on the other hand, constitute portions of embodiments of the present invention.

VI.B. Summary of helper processes

As described above, the helper processes comprise the steps of: preprocessing data; scheduling; loading a model; and adding ratings.

VI.B.1.Preprocessing data

Client data may exist in a variety of possible formats, any one of which may not be directly usable by the system. Furthermore, there may be multiple data sources that collectively embody the "ratings data" or the "sparse ratings data." This data must be converted to a format that is suitable for the system, as indicated by preprocessing box 55 in FIG. 3, so that further processing steps may use this data in its compressed sparse representation. Section XII.C below discusses the general forms data may take in more detail.

VI.B.2.Scheduling

In a preferred embodiment of the present invention, scheduler 40, as indicated schematically in FIG. 3, is a functional unit of the present invention that enables processing to be initiated after which the instantiation of the system occurs.

VI.B.3.Loading a model and adding rating(s)

Because of a separation between on-line model processing (or runtime processing) and off-line model processing in various embodiments of the present invention, one skilled in the art should appreciate the step of loading the runtime model used in on-line processing. This is indicated schematically by runtime model loader 405 in FIG. 3. Furthermore, the ability to add additional ratings to the models described by the present invention is incorporated in its design and is indicated schematically as add ratings box 65 in FIG. 3. In one preferred embodiment of the present invention, the additional ratings may originate as additional data from the on-line or runtime processing region 400. Furthermore, the arrow connecting ratings matrix data 170 and personal recommendation 310 indicates that, in certain instances, personal recommendations may be directly implemented from ratings matrix data 170.

FIG. 4 depicts a system configuration consistent with the present invention in which runtime recommendation system 550 cooperates with off-line recommendation system 520 over network 510. Both runtime recommendation system 550 and off-line recommendation system 520 include processors as well as memory. In particular, off-line recommendation system 520 includes memory 540 for the storage of sparse matrix 545 information, and memory 530 for the storage of rules 535 for off-line processing. Likewise, runtime recommendation system 550 includes memory 560 for the storage of runtime model 560 and memory 570 for the storage of rules 575 for runtime processing. In a preferred embodiment runtime recommendation system 550 may form a portion of a data processing device with conventionally limited memory capabilities such as a personal digital assistant (PDA) or a mobile phone. In practice, however, one skilled in the art will appreciate that runtime recommendation system 550 may form a part of any data processing device such as a personal computer, workstation, or mainframe. Furthermore, the depiction of network 510 between runtime recommendation system 550 and off-line recommendation system 520 is exemplary only. That is, one skilled in the art should appreciate that runtime recommendation system 550 and off-line recommendation system 520 may form different processing and memory portions of the same data processing device.

VII. FIRST EMBODIMENT OF THE PRESENT INVENTION

In a first embodiment of the present invention, depicted schematically in FIG. 5, a method of providing a recommendation to a user comprises: providing a sparse ratings matrix, forming a plurality of data structures representing the sparse ratings matrix, forming a runtime recommendation model from the plurality of data structures, determining a recommendation from the runtime recommendation model in response to a request from a user, and providing the recommendation to the user.

In one example, discussed below, the plurality of data structures corresponds to the partitioning of a ratings matrix into a plurality of sub-space matrices, where one of the plurality of sub-space matrix is manipulated either singly or with a second sub-space matrix to produce a recommendation. For example, the plurality of sub-space ratings matrix may correspond to a plurality of categories. In FIG. 3, this is schematically depicted as initialize box 150, and in FIG. 5, this corresponds to step 610.

In general, step 605 of FIG. 5 includes retrieving or otherwise receiving data corresponding to a sparse ratings matrix. As mentioned above, step 610 initializes the sparse ratings matrix for further processing by forming a plurality of data structures representing the sparse ratings matrix.

In initializing sparse matrices of the present invention (step 610), the off-line model creation as described herein is based on computing the products $\boldsymbol{R}^t\boldsymbol{R}$ and $\boldsymbol{R}\boldsymbol{R}^t$. Below, it is shown that the task of model creation may be decomposed into constituent building blocks that are computed from matrix products.

VII.A. Categorical Data

As used herein, *categories* are mappings between dimensions. As previously discussed, the process of making recommendations for clients using one dimension's ratings for another dimension's recommendations concerned a situation in which there are ratings for both dimensions. Suppose that this is not the case, and that one only has ratings data for dimension *i*, but that one has a mapping between dimension *i* and *j*. For example, dimension *j* may represent categories for items contained in dimension *i*. Let this mapping, denoted by $\boldsymbol{T}$, be given by

$$\boldsymbol{T} = \boldsymbol{T}_{i,c} = \begin{cases} 1 : \text{if item } i \text{ is contained in category } c \\ 0 : \text{otherwise} \end{cases}$$

If matrix ***R*** is multiplied on the right by ***T***, the resulting matrix may be considered a rating matrix of clients to categories. It is interesting to note that the scale should now be considered an interest scale as discussed in more detail below in Section XII.C.1, because higher valued entries denote that the respective client rated more items in this category than in category entries with lower values.

However, the discussion in this Section is concerned with the issue of mapping one's ability to make recommendations in one dimension to making recommendations in another in which there is no ratings data. For this, the model is defined by

$$_{(I\text{-}I)}\boldsymbol{M} = \boldsymbol{R^t} \bullet \boldsymbol{R} \bullet \boldsymbol{T}$$

and one possible recommendation model is given by (step 615)

$$\textbf{Unary\_Multi\_Vote}(i,c)_{k'}(\boldsymbol{R}_{u,*},k) = \textbf{TOP}_{k'}\Big(\sum_{z \in \boldsymbol{R}_{u,*}} {}_{(I\text{-}I)}\boldsymbol{M}(i,c)^{r(k)}{}_{z,*}\Big)$$

In this instance, ***T*** induces a ratings matrix for dimension *c*. This result suggests another approach for making recommendations in this case (step 630) by letting

$$_{(I\text{-}I)}\boldsymbol{M} = \boldsymbol{R^t} \bullet \textbf{Unary}(\boldsymbol{R} \bullet \boldsymbol{T})$$

in step 615.

VII.B. <u>Distributed Modeling</u>

In this section, the following temporal concepts are discussed: (1) **Model Creation (MC)**: the point in time in which the model is created; (2) **Runtime Model Creation (RMC)**: the point in time in which the runtime model is created; (3) **Request Recommendation (RR)**: the point in time in which the recommendation is requested;

and (4) **Recommendation Process (RP)** the point in time in which the recommendation is processed. From the above itemization of concepts, the "Recommendation Process" is essentially the final modification of the recommendation model (step 630). In a preferred embodiment of the present invention the temporal ordering among these four events is such that each event preferably occurs before the next may be considered for processing. However, such ordering is not a requirement of the present invention.

Further still, **Personalization Identifiers (PI)** are the key indicators that allow personalization to begin. Before the personalization identifiers are known, it is preferable to have as much model creation as is possible occur. After the PIs exist and a recommendation request occurs (at runtime), the final recommendation process may begin. In addition, **Personalization Identification (PIndent)** is the time at which the personalization identifiers have become known.

One aspect of these definitions is that in scenarios in which personalization identification occurs before the request for personalization, there is an opportunity for efficiently pre-calculating derived models that are customized to the fact that personalization identification has occurred. This provides an opportunity for extremely detailed personalization. In addition, the following list indicates areas in which distributed computed techniques may be involved: (i) parallel processing of model creation and various derived models; (ii) parallel processing of the recommendation process (i.e., the construction of the final recommendation); (iii) business-to-business model sharing; and (iv) efficient calculation of the entire model, together with distributing derived models to locations where the local recommendation are made. Specific implementations of distributed model creation and manipulation are discussed in detail below.

VII.C. Distributing Model Creation

As discussed earlier regarding ratings matrices based in different dimensions:

$$\boldsymbol{R}^{(1)},\boldsymbol{R}^{(2)},\ldots,\boldsymbol{R}^{(p)}$$

formulas were presented that described the calculation required to compute both the complete co-rate model or the runtime co-rate models corresponding to either $\boldsymbol{R}^{t}\boldsymbol{R}$ or $\boldsymbol{R}\boldsymbol{R}^{t}$. Since the ratings matrix $\boldsymbol{R}$ may be banded or striped, as discussed below, the calculation from that section yields a manner in which to distribute the calculation of these models, as, for example, in step 615 of FIG. 5.

VII.C.1. Banding by Rows

Consider a partition of all clients for which there exists a item rating. Denote this partition as $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_k)$ and define the *bands*

$$\boldsymbol{R}^{(j)}_{u,i} = \begin{cases} \boldsymbol{R}_{u,i} : \text{if } u \in \boldsymbol{B}_j \\ 0 : \text{otherwise} \end{cases}$$

Since the bands are partitioning the rating by clients, it will be simple to derive a update formula directly. The result is a special case of a derivation provided below. Consider

$$\boldsymbol{R} = \boldsymbol{R}^{(1)} + \boldsymbol{R}^{(2)} + \ldots + \boldsymbol{R}^{(k)}$$

where $\boldsymbol{R}^{(j)}$ is defined above. The model is given by the following calculation

$$\boldsymbol{R}^t \boldsymbol{R} = (\sum_{i=1}^{k} \sum_{j=1}^{k} (\boldsymbol{R}^{(i)})^t)(\boldsymbol{R}^{(j)})$$

Collecting terms and reordering the summations gives

$$\boldsymbol{R}^t \boldsymbol{R} = (\sum_{i=1}^{k} \sum_{j=1}^{k} (\boldsymbol{R}^{(i)})^t)(\boldsymbol{R}^{(j)})$$

$$= \sum_{i=1}^{k} \left\{ (\boldsymbol{R}^{(i)})^t \boldsymbol{R}^{(i)} + \sum_{j:i<j\leq k} \left\{ \boldsymbol{R}^{(i)}, \boldsymbol{R}^{(j)} \right\} \right\}$$

where the internal summation is over terms that are all zero. Hence derivation reduces to

$$\boldsymbol{R}^t \boldsymbol{R} = \sum_{i=1}^{k} \left\{ (\boldsymbol{R}^{(i)})^t \boldsymbol{R}^{(i)} \right\}$$

This formula implies that, banded by clients, the co-rate model $\boldsymbol{R}^t \boldsymbol{R}$ may be distributed to multiple nodes of a computing cluster. Each node of such a cluster can compute its respective piece of the model. After any two nodes of such a cluster have computed their portion(s) of the model, communication to add the terms computed is permissible, etc. If there are 2^n bands of clients, the final model may be constructed with as few as n parallel steps, with a total of $2^n - 1$ summations. Powers of two have been chosen for convenience. This will now be clarified by the following recursion formula. Let the base of the recursion be defined by

$$_{(I\text{-}I)}\boldsymbol{M}_i(0) = (\boldsymbol{R}^{(i)})^t \boldsymbol{R}^{(i)} \text{ where } j : 1 < j \leq 2^n$$

the recursion is now given by

$$_{(I\text{-}I)}\boldsymbol{M}_j(k) = \boldsymbol{M}_{2j-1}(k-1) + \boldsymbol{M}_{2j}(k-1) \text{ where } j : 1 \leq j \leq 2^{n-k} \text{ and } k : 1 \leq k \leq n$$

When $k = n, j$ can only take the value 1, at which point the recursion terminates and

$$(I\text{-}I)\quad \boldsymbol{M}_1(n) = \boldsymbol{R}^t \boldsymbol{R}$$

The distribution of the calculation is made easier due to the cross-band terms reducing to zero. In the next section this will not be the case.

VII.C.2. Striping by Columns

In this section, the ratings matrix will be striped by items instead of banded by clients. The motivation for this approach is to calculate extremely large data sets in which both $\boldsymbol{R}^t\boldsymbol{R}$ and $\boldsymbol{R}\boldsymbol{R}^t$ are to be calculated. Striping by items is preferred for the $\boldsymbol{R}^t\boldsymbol{R}$ case (and by client for the $\boldsymbol{R}\boldsymbol{R}^t$ case). In either case, striping as opposed to banding causes the cross-stripe terms be non-zero, requiring some discussion of distributing the calculation to combining intermediate results to calculate the final result. Many of the previous results still apply and will be the starting point for this discussion.

Consider that there are N blocks enumerated from 0 to $N-1$ as $\boldsymbol{B} = (\boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_{N-1})$ with the following definition:

$$\boldsymbol{R}^{(j)}_{u,i} = \begin{cases} \boldsymbol{R}_{u,i} : \text{if } i \in \boldsymbol{B}_j \\ 0 : \text{otherwise} \end{cases}$$

Similar to before, one has

$$\boldsymbol{R} = \boldsymbol{R}^{(1)} + \boldsymbol{R}^{(2)} + \ldots + \boldsymbol{R}^{(N-1)}$$

By reorganizing the terms

$$R^t R = \sum_{i=1}^{N-1} \left\{ (R^{(i)})^t R^{(i)} + \sum_{j:i<j\leq N-1} \left\{ R^{(i)}, R^{(j)} \right\} \right\}$$

$$= \sum_{i=1}^{N-1} (R^{(i)})^t R^{(i)} + \sum_{i=1}^{N-1} \sum_{j:i<j\leq N-1} \left\{ R^{(i)}, R^{(j)} \right\}$$

$$= \sum_{i=1}^{N-1} (R^{(i)})^t R^{(i)} + \sum_{(i,j):0\leq i<j\leq N-1} \left\{ R^{(i)}, R^{(j)} \right\}$$

Therefore, one finds that the previous recursion applies to the first summation, which leaves distributing the second summation. Reordering the second summation yields

09887528.052501

$$\sum_{(i,j):0\leq i<j\leq N-1} \left\{ R^{(i)}, R^{(j)} \right\} = \sum_{k=1}^{N-1} \sum_{\substack{0\leq i,j\leq N-1 \\ j-i=k}} \left\{ R^{(i)}, R^{(j)} \right\}$$

$$= \sum_{k=1}^{N-1} \sum_{i=0}^{N-k-1} \left\{ R^{(i)}, R^{(i+k)} \right\}$$

$$= \sum_{k=1}^{N-1} \sum_{i=0}^{N-k-1} (i,i+k)$$

where

$$(i,j) = \begin{cases} \left\{ R^{(i)}, R^{(j)} \right\} & : \text{if } i<j \\ \left\{ R^{(i)}, R^{(j)} \right\} & : \text{if } i>j \\ 0 & : \text{otherwise} \end{cases}$$

In this summation, only the case $i < j$ is realized, but the notation will be useful in the next section. If the computing cluster has N nodes, and $R^{(i)}$ is stored on the ith node, then the above summation indicates which nodes communicate to complete the computation of the model.

VII.C.3. An Example of Distributed Model Creation Striped by Columns

Consider a four-node computing cluster for which there are 16 stripes required to represent ***R***. Tables A1 and A2 indicate the distribution of work in the case of 16 nodes. In this scenario, half the of the nodes of the cluster are idle in eight stages.

Table A1

Stage	Node 0	node 1	node 2	node 3	node 4	node 5	node 6	node 7
-1-	(0,1)	(1,2)	(2,3)	(3,4)	(4,5)	(5,6)	(6,7)	(7,8)
-2-	(0,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	(6,8)	(7,9)
-3-	(0,3)	(1,4)	(2,5)	(3,6)	(4,7)	(5,8)	(6,9)	(7,10)
-4-	(0,4)	(1,5)	(2,6)	(3,7)	(4,8)	(5,9)	(6,10)	(7,11)
-5-	(0,5)	(1,6)	(2,7)	(3,8)	(4,9)	(5,10)	(6,11)	(7,12)
-6-	(0,6)	(1,7)	(2,8)	(3,9)	(4,10)	(5,11)	(6,12)	(7,13)
-7-	(0,7)	(1,8)	(2,9)	(3,10)	(4,11)	(5,12)	(6,13)	(7,14)
-8-	(0,8)	(1,9)	(2,10)	(3,11)	(4,12)	(5,13)	(6,14)	(7,15)

Table A2

Stage	node 8	node 9	node 10	node 11	node 12	node 13	node 14	node 15
-1-	(8,9)	(9,10)	(10,11)	(11,12)	(12,13)	(13,14)	(14,15)	(15,0)
-2-	(8,10)	(9,11)	(10,12)	(11,13)	(12,14)	(13,15)	(14,0)	(15,1)
-3-	(8,11)	(9,12)	(10,13)	(11,14)	(12,15)	(13,0)	(14,1)	(15,2)
-4-	(8,12)	(9,13)	(10,14)	(11,15)	(12,0)	(13,1)	(14,2)	(15,3)
-5-	(8,13)	(9,14)	(10,15)	(11,0)	(12,1)	(13,2)	(14,3)	(15,4)
-6-	(8,14)	(9,15)	(10,0)	(11,1)	(12,2)	(13,3)	(14,4)	(15,5)
-7-	(8,15)	(9,0)	(10,1)	(11,2)	(12,3)	(13,4)	(14,5)	(15,6)
-8-	No-Op	No-Op	No-Op	No-Op	No-Op	No-Op	No-Op	No-Op

In the alternative, one may reorganize the terms to be calculated for computation on a four-node cluster as indicated below in Tables B1 and B2. Note, however, that the partitioning presented is not the only way to partition the calculation.

Table B1

Stage	node 0	node 1	node 2	node 3
-1-	(0,1)	(1,2)	(2,3)	(3,4)
-2-	(0,2)	(1,3)	(2,4)	(3,5)
-3-	(0,3)	(1,4)	(2,5)	(3,6)
-4-	(0,4)	(1,5)	(2,6)	(3,7)
-5-	(0,5)	(1,6)	(2,7)	(3,8)
-6-	(0,6)	(1,7)	(2,8)	(3,9)
-7-	(0,7)	(1,8)	(2,9)	(3,10)
-8-	(0,8)	(1,9)	(2,10)	(3,11)
-9-	(8,9)	(9,10)	(10,11)	(11,12)
-10-	(8,10)	(9,11)	(10,12)	(11,13)
-11-	(8,11)	(9,12)	(10,13)	(11,14)
-12-	(8,12)	(9,13)	(10,14)	(11,15)
-13-	(8,13)	(9,14)	(10,15)	(11,0)
-14-	(8,14)	(9,15)	(10,0)	(11,1)
-15-	(8,15)	(9,0)	(10,1)	(11,2)
-16-	No-Op	No-Op	No-Op	No-Op

Table B2

stage	node 0	node 1	node 2	node 3
-17-	(4,5)	(5,6)	(6,7)	(7,8)
-18-	(4,6)	(5,7)	(6,8)	(7,9)
-19-	(4,7)	(5,8)	(6,9)	(7,10)
-20-	(4,8)	(5,9)	(6,10)	(7,11)
-21-	(4,9)	(5,10)	(6,11)	(7,12)
-22-	(4,10)	(5,11)	(6,12)	(7,13)
-23-	(4,11)	(5,12)	(6,13)	(7,14)
-24-	(4,12)	(5,13)	(6,14)	(7,15)
-25-	(12,13)	(13,14)	(14,15)	(15,0)
-26-	(12,14)	(13,15)	(14,0)	(15,1)
-27-	(12,15)	(13,0)	(14,1)	(15,2)
-28-	(12,0)	(13,1)	(14,2)	(15,3)
-29-	(12,1)	(13,2)	(14,3)	(15,4)
-30-	(12,2)	(13,3)	(14,4)	(15,5)
-31-	(12,3)	(13,4)	(14,5)	(15,6)
-32-	No-Op	No-Op	No-Op	No-Op

FIG. 6 depicts a conventional node communication pattern. Within each box at a given stage the term being computed is indicated. The number at the tail of each arrow indicates the block being passed at that stage. FIG. 6 also indicates some general properties that are useful in characterizing the general case. In order to precisely describe the processing, one may make the following definitions. Let k denote the number of nodes in the computing cluster and N be the number of stripes of $\boldsymbol{R}$ such that $2k$ divides

N. Defining the permutation matrix for the shift operator as

$$(\Theta_{(k)})_{i,j} = \begin{cases} 1 : \text{if } j \geq i \quad j - i = 1 \quad 0 \leq i, j \leq k-1 \\ 1 : \text{if } j < i \quad i - j = k - 1 \quad 0 \leq i, j \leq k-1 \\ 0 : \text{otherwise} \end{cases}$$

the n^{th} power of $\Theta_{(k)}$ is given by the following

$$((\Theta_{(k)})^{n})_{i,j} = \begin{cases} 1 : \text{if } j = (n\,\mathbf{Mod}_k + i)\,\mathbf{Mod}_k \quad 0 \leq i,j \leq k-1 \\ 0 : \text{otherwise} \end{cases}$$

In general, blocks being received by the nodes at stage n are given by $(\Theta_k)^n$ multiplied by a vector which characterizes the blocks sent at stage n. The block number that node i sends at stage n for a cluster size of k is denoted by **Send_Block**(n; i; k) and is given functionally by

$$\textbf{Send\_Block}(n; i; k) = \begin{cases} \Bigg(\Bigg(\Bigg(\left\lfloor \frac{(n-i-1)\,\mathbf{Mod}_N}{k} \right\rfloor + \left\lceil \frac{n}{N} \right\rceil \Bigg) \cdot k + i \Bigg)\,\mathbf{Mod}_N : \text{if } 0 \neq n\,\mathbf{Mod}_N \geq i \\ \left\lfloor \frac{n}{N} \right\rfloor \cdot k + i \qquad : \text{if } 0 \neq n\,\mathbf{Mod}_N < i \\ \varnothing \qquad : 0 = n\,\mathbf{Mod}_N \end{cases}$$

The received bands are characterized by

Received_Block(n; i; k) = $(\Theta_k)^n$ **Send_Block**(n; i; k)

Note that the functional form of $(\Theta_{(k)})^n$ gives the communication required at stage n for each node of a computing cluster with k nodes. As such, the function **Received_from**(n; i; k) is defined by

$$\textbf{Received\_from}(n; i; k) = (\, n\ \textbf{Mod}_k + i)\ \textbf{Mod}_k$$

and reducing **Received_Block**(n; i; k) gives

$$\textbf{Received\_Block}(n; i; k) = \textbf{Send\_Block}(n; \textbf{Received\_from}(n; i; k); k)$$

$$\textbf{Current\_Block}(n; i; k) = \left\lfloor \frac{2((n-1)\ \textbf{Mod}_N)}{N} \right\rfloor \bullet \frac{N}{2} + \left\lfloor \frac{n}{N} \right\rfloor \bullet k + i$$

$$\textbf{Send\_to}(n; i; k) = (k - n\ \textbf{Mod}_k + i)\ \textbf{Mod}_k$$

Putting partial calculations together, it is now possible to express functionally the calculation at the i^{th} node at the n^{th} stage. In what follows, k and N are suppressed. Let

$$\boldsymbol{rb}_i(n) = \textbf{Received\_Block}(n; i; k)$$

$$\boldsymbol{cb}_i(n) = \textbf{Current\_Block}(n; i; k)$$

$$\textbf{Current\_Block\_Matrix}(n : i) = (\boldsymbol{cb}_i(n), \boldsymbol{rb}_i(n))$$

$$= \begin{cases} \left\{ \boldsymbol{R}(\boldsymbol{cb}_i(n)), \boldsymbol{R}(\boldsymbol{rb}_i(n)) \right\} & : \text{if } \boldsymbol{cb}_i(n) < \boldsymbol{rb}_i(n) \\ \left\{ \boldsymbol{R}(\boldsymbol{rb}_i(n)), \boldsymbol{R}(\boldsymbol{cb}_i(n)) \right\} & : \text{if } \boldsymbol{cb}_i(n) > \boldsymbol{rb}_i(n) \\ 0 & : \text{otherwise} \end{cases}$$

Accordingly, the process of distributing model creation is easily implemented consistent with the present invention.

VII.C.4. Distributed Computing for Recommendation Processing

It is also possible not to add all the pieces together and make recommendation directly from the nodes. That is, it is possible to have node i keep the portion of the runtime model for the items in $\boldsymbol{R}^{(i)}$. In a preferred embodiment, each node must communicate with the other nodes to compute the proper portion of the model. This is given by

$$M_{node_i} = \sum_{k=-i+1}^{2^n - i} (\boldsymbol{R}^{(i)})^t \boldsymbol{R}^{(i+k)}$$

The run-time distributed model (step 615) is given by

$$M_{node_i}^{r(k)} = \textbf{Unary}_k(M_{node_i})$$

Furthermore, let $\boldsymbol{S}$ be a set of items from which a recommendation will be processed. If one partitions this set of items according to with nodes representing the items of $\boldsymbol{S}$, then $\boldsymbol{S} = \cup_{1 \le i \le 2^n} \boldsymbol{S}_i$, where $\boldsymbol{S}_i$ is the restriction of $\boldsymbol{S}$ to the i^{th} node. Since the union is disjoint, the voting algorithm may be distributed according to the formula

$$\textbf{Unary\_Multi\_Vote}_k(S, k) = \textbf{TOP}_{k'}\left(\sum_{1 \le j \le 2^n} \sum_{i \in \boldsymbol{S}_j}{}' \left(M_{node_j}^{r(k)} \right)_{i,*} \right)$$

Accordingly, the results from a combination of nodes may be efficiently processed consistent with the present invention.

VII.D. Business-to-Business Model Sharing

Next, one can consider either different divisions of a company or different companies that identify a common set of clients from which to build cross-company or cross-division co-rate models. This is straightforward given the method developed for parallel dimension and distributed computing above. The only issue is one of agreement on a set of clients.

VII.E. Mobilized Distributed Personalization

Having described the methods for calculating models in their entirety—by computing pieces of models and computing derived models—the next task focuses on distributing the model to localities at which recommendation processing will occur. Models may be distributed in their entirety if memory constraints permit. More to the point, by having pre-processed the models it is now feasible to distribute only portions of the model that are usable by a given set of personalization identifiers (*i.e.*, a single client's ratings). It is shown below that a personalized model maybe produced, thereby further personalizing the client's experience. In exactly the same fashion that a model may be partitioned for model creation and the recommendation process, a client's ratings may be partitioned according to categories.

Suppose that there are q categories. One may denote the list of categories as $\boldsymbol{C} = (\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots ,\boldsymbol{C}_q)$. Each $\boldsymbol{C}_j$ is a list of items consisting of the members of the j^{th} category. Since $\boldsymbol{C}_j$ is a set of items, it may be considered a sparse vector with values of ones. As such $\boldsymbol{D}(\boldsymbol{C}_j)$ is the diagonal matrix with ones on the diagonal corresponding to the list $\boldsymbol{C}_j$. Let (step 610)

$$\boldsymbol{R}^{\boldsymbol{C}_j} = \boldsymbol{R} \bullet \boldsymbol{D}(\boldsymbol{C}_j)$$

The model for co-rating all items to category $\boldsymbol{C}_j$ is given by (step 615)

$$M(\boldsymbol{C}_j)^{r(k)} = \mathbf{Unary}_k\,(\boldsymbol{R}^t\boldsymbol{R}^{\boldsymbol{C}_j})$$

where j runs over the list of categories $\boldsymbol{C}$. Suppose that one wants to specialize these models to a client's rating. This may be useful because one is going to distribute the model. Consider starting with $\boldsymbol{D}(\boldsymbol{R}_{u,*})$, the diagonal matrix corresponding to client u's ratings. In a preferred embodiment, it will be customary to personalize the models $M^{r(k)}\ (\boldsymbol{C}_j)$ to a client's ratings given by the non-zero rows of (step 615)

$$M^{u;r(k)}(\boldsymbol{C}_j) = \boldsymbol{D}(\boldsymbol{R}_{u,*}) \bullet M^{r(k)}(\boldsymbol{C}_j)$$

For example, suppose "restaurants" is a category, then a model for recommending restaurants from client u is given by$M_u^{u;r(k)}\ (\boldsymbol{C}_{resturants})$. Since the personalized model is produced by multiplying by a sparse diagonal matrix, the number of non-zero rows in question should be significantly smaller than that of the respective model. As a result, the collection of personalized models across all categories may be a reasonably sized set. One may define the block matrix representing these derived models by

$$M^{u;r(k)}(\boldsymbol{C}) = \begin{pmatrix} M^{u;r(k)}(\boldsymbol{C}_1) \\ M^{u;r(k)}(\boldsymbol{C}_2) \\ \bullet \\ \bullet \\ \bullet \\ M^{u;r(k)}(\boldsymbol{C}_q) \end{pmatrix}$$

One may also consider the matrix that characterizes a client's rating across the list of categories $\boldsymbol{C}$ denoted by

$$\boldsymbol{R}_u(C) = \begin{pmatrix} \boldsymbol{R}_{u,*} | C_1 \\ \boldsymbol{R}_{u,*} | C_2 \\ \bullet \\ \bullet \\ \bullet \\ \boldsymbol{R}_{u,*} | C_q \end{pmatrix}$$

09887528.062501

Although $\boldsymbol{R}_u(\boldsymbol{C})$ and $M^{u;r(k)}(\boldsymbol{C})$ are somewhat complicated to express formally, in practice the amount of represented data is relatively small as compared to the entire model. These sets correspond to the personalized portions of the model with respect to client u and the category partitioning induced by $\boldsymbol{C}$. Making j-recommendations from a unary multiplicity voting algorithm using client u's z-ratings may be given by

$$\textbf{Unary\_Multi\_Vote}_k(\boldsymbol{R}_u(C)_z, k) = \textbf{TOP}_{k'}\left(\sum_{i \in \boldsymbol{R}_u(C)_j}{}' \; M_{i,*}^{u;r(k)}(\mathrm{C}_j) \right)$$

Of course, there are other meaningful expressions for making such recommendations if the co-rate values are taken into consideration. Furthermore, it is also consistent with the present invention to employ a client's entire set of ratings to make recommendations into a category, and if one of the categories is the entire item set then this will, in fact, be the case. This formulation enables recommendations to be made from a client's ratings in either a specific category or across categories. In all cases, if the client's personalization identities are known prior to the request for recommendation, the respective parts of the personalized model may be calculated and distributed to the location where the recommendations will be processed. This might be a desktop unit, cellular phone, personal digital assistant, digital assistant in a car, or some other device.

VII.E.1. A scenario for personalization using personal digital assistants

Further still, as an example, consider the following scenario. A person on a business trip carries a personal digital assistant (PDA). The PDA may or may not have Internet connectivity. If Internet connectivity is not present, then it is assumed that the person has some other means of Internet connectivity. Suppose that a database exists that contains this person's ratings for restaurants, theater, recreation, clothing, shopping, etc. It would be useful to make recommendations for any of these categories from this person's ratings of such categories. As an example, it makes no sense to recommend a restaurant in a different city. On the other hand, it is of utility to load derived models, such as those described in this section, directly into the clients PDA. This may be achieved by the PDA's direct Internet connectivity or by some other means. In either case, the models needed to make personalized recommendations for this client may be restricted to a memory allocation size to allow the entire footprint to fit into the PDA. In such a case, the algorithms of this and other sections are of a sufficiently simple nature that computation on the PDA is possible. If the PDA does in fact have connectivity, then an update of its internal models is possible on an ongoing basis. Otherwise, it could be updated out-of-band. Although out-of-band updating is less responsive than that of in-band updating, the rate at which added ratings will change the outcome of the recommendations is a minor effect in many scenarios. In this manner, a PDA could have the required runtime capability to make recommendations personalized to a client's ratings and derived personalized models. Thus, recommending breakfast in a particular city may be personalized depending on what one had for dinner the night before.

VIII. SECOND EMBODIMENT OF THE PRESENT INVENTION

In a second embodiment of the present invention, depicted in FIG. 7, a method for providing a recommendation to a client is based upon a model that is incrementally updated. Within FIG. 3, this corresponds to the process indicated by update box 140, and in FIG. 7 this corresponds to step 720.

A fresh model for on-line (or runtime) processing is required in order to make accurate recommendations. Whether the model is computed from scratch or the model is incrementally updated, the operations performed reduce to efficiently performing sparse matrix operations.

VIII.A. Incrementally Updating Model and Model Creation

Updating co-rates is a means by which a recommendation model is incrementally updated in a fashion analogous to the fundamental theorem of integral calculus. Specifically, and as discussed earlier, the model for making recommendations comes from the matrix of co-rated items ${}_{(I\text{-}I)}\mathbf{M} = \mathbf{R}^{t}\mathbf{R}$. For now, consider that the number of items is fixed, resulting in ${}_{(I\text{-}I)}\mathbf{M}$ being of fixed dimension. As time evolves, the matrix ${}_{(I\text{-}I)}\mathbf{M}$ changes and it is natural to consider the matrix to be a function of time, denoted ${}_{(I\text{-}I)}\mathbf{M}(t)$. Imagine that one could instantaneously compute the matrix of co-rated items. Clearly, this would be ideal. As new ratings arrive, the co-rate matrix jumps at the time of updating, first taking its initial value and then jumping to its next value, and so on. Note that the fixed number of items results in a model of bounded dimension. This is not general enough for the purposes of the present invention and as such the underlying state space for $\boldsymbol{R}$ should be extended to be infinite by infinite matrices indexed by the non-negative integers by the non-negative integers, with the property that only a finite number of clients and items have non-zero entries. For example, $\boldsymbol{R}_{ui}$ is really a doubly indexed array, starting at zero for each of the independent coordinates. All that one is now allowing is that the arrays in question are finite and not of fixed dimension. The extension requires that our methods for update work independently of the number of clients and items. Since the multiplication of two matrices from our extended state space yields a matrix in the state space, the methods will function smoothly.

Note that in a time interval [0, T), there are only a finite number of jumps and therefore the jump sequence may be enumerated as $t_0 = 0 < t_1 < t_2 < t_3 < \ldots < t_n < T$. At any time in [0, T), the model's change is given by

$$\delta\left[{}_{(I\text{-}I)}\boldsymbol{M}(t)\right] = {}_{(I\text{-}I)}\boldsymbol{M}(t) - {}_{(I\text{-}I)}\boldsymbol{M}(t^-) = \boldsymbol{R}^t(t)\,\boldsymbol{R}(t) - \boldsymbol{R}(t^-)\boldsymbol{R}(t^-)$$

where t^- indicates the use of a left limit. In other words, if there is a jump at time t, then $\boldsymbol{R}(t^-)$ is the ratings matrix the moment just before the update occurred. If t is not in the jump sequence, then $\delta[{}_{(I\text{-}I)}\boldsymbol{M}(t)]$ equals the matrix of all zeros. Since the jump times are discrete, then, for two consecutive jump times one, has the relationship $\boldsymbol{R}(t_j) = \boldsymbol{R}(t^-_{j+1})$. This is nothing more than a restatement that $\boldsymbol{R}$ is constant between updates. Let time $T > 0$ be chosen and as above enumerate the times at which jumps occur. We can express the matrix of co-rates as a telescoping sum of incremental updates, given by, in general, (considering both ${}_{(I\text{-}I)}\boldsymbol{M}(t)$ and ${}_{(c\text{-}c)}\boldsymbol{M}(t)$)

$$\boldsymbol{M}(T) = \boldsymbol{M}(t_n) = (\boldsymbol{M}(0) - \boldsymbol{M}(0^-)) + (\boldsymbol{M}(t^1) - \boldsymbol{M}(t^-_1)) + \ldots + (\boldsymbol{M}(t_n) - \boldsymbol{M}(t^-_n))$$

As a reasonable convention, one may choose $\boldsymbol{M}(0^-)$ to be the matrix of zeros. Letting $t_0 = 0$, $\boldsymbol{M}(T)$ may be expressed in the following summation (step 720 of FIG. 7):

$$\boldsymbol{M}(T) = \sum_{j=0}^{n} \delta\boldsymbol{M}(t_j)$$

$$= \boldsymbol{M}(t_0) + \sum_{j=1}^{n} \delta\boldsymbol{M}(t_j)$$

$$= \boldsymbol{M}(0) + \sum_{j=1}^{n} \delta\boldsymbol{M}(t_j)$$

This expression is a discrete analogue to the Fundamental Theorem of Integral Calculus. Accordingly, one may return to $\delta[_{(I-I)}\boldsymbol{M}(t)]$ and compute a product rule for the increment:

$$\delta\left[_{(I-I)}\boldsymbol{M}(t)\right] = \boldsymbol{R}^t(t)\boldsymbol{R}(t) - \boldsymbol{R}^t(t^-)\boldsymbol{R}(t^-)$$

$$= \boldsymbol{R}^{\boldsymbol{t}}(t)\boldsymbol{R}(t) - \boldsymbol{R}^{\boldsymbol{t}}(t)\boldsymbol{R}^{\boldsymbol{t}}(t^-) + \boldsymbol{R}^{\boldsymbol{t}}(t)\boldsymbol{R}^{\boldsymbol{t}}(t^-) - \boldsymbol{R}^{\boldsymbol{t}}(t^-)\boldsymbol{R}(t^-)$$

$$= \boldsymbol{R}^{\boldsymbol{t}}(t)(\boldsymbol{R}(t) - \boldsymbol{R}^{\boldsymbol{t}}(t^-)) + (\boldsymbol{R}^{\boldsymbol{t}}(t) - \boldsymbol{R}^{\boldsymbol{t}}(t^-))\boldsymbol{R}(t^-)$$

$$= \boldsymbol{R}^{\boldsymbol{t}}(t)\delta\boldsymbol{R}(t) + \delta\boldsymbol{R}^{\boldsymbol{t}}(t)\boldsymbol{R}(t^-)$$

$$= \boldsymbol{R}^{\boldsymbol{t}}(t)\delta\boldsymbol{R}(t) + [\delta\boldsymbol{R}(t)]^{\boldsymbol{t}}\boldsymbol{R}(t^-)$$

$$= [\boldsymbol{R}(t^-) + \delta\boldsymbol{R}(t)]^{\boldsymbol{t}}\delta\boldsymbol{R}(t) + [\delta\boldsymbol{R}(t)]^{\boldsymbol{t}}\boldsymbol{R}(t^-)$$

$$= [\boldsymbol{R}(t^-)]^{\boldsymbol{t}}\delta\boldsymbol{R}(t) + [\delta\boldsymbol{R}(t)]^{\boldsymbol{t}}\delta\boldsymbol{R}(t) + [\delta\boldsymbol{R}(t)]^{\boldsymbol{t}}\boldsymbol{R}(t^-)$$

$$= [\boldsymbol{R}(t^-)]^{\boldsymbol{t}}\delta\boldsymbol{R}(t) + [\delta\boldsymbol{R}(t)]^{\boldsymbol{t}}\boldsymbol{R}(t^-) + [\delta\boldsymbol{R}(t)]^{\boldsymbol{t}}\delta\boldsymbol{R}(t)$$

$$= [\boldsymbol{R}(t^-)]^{\boldsymbol{t}}\delta\boldsymbol{R}(t) + [[\boldsymbol{R}(t^-)]^{\boldsymbol{t}}\delta\boldsymbol{R}(t)]^{\boldsymbol{t}} + [\delta\boldsymbol{R}(t)]^{\boldsymbol{t}}\delta\boldsymbol{R}(t)$$

$$= \boldsymbol{R}^{t}(t^{-})\delta\boldsymbol{R}(t) + [\boldsymbol{R}^{t}(t^{-})\delta\boldsymbol{R}(t)]^{t} + [\delta\boldsymbol{R}(t)]^{t}\delta\boldsymbol{R}(t)$$

Note that $\delta[{}_{(I\text{-}I)}\boldsymbol{M}(t)]$ is a symmetric matrix and only involves the values of $\boldsymbol{R}(t^{-})$ (the sparse matrix of step 705) and $\delta\boldsymbol{R}(t)$ (the update ratings matrix of step 710). Although it appears that all of $\boldsymbol{R}(t^{-})$ is being used, this is not necessary. We only need to extract the client rows that can contribute to $\delta\boldsymbol{R}(t^{-})$.

Letting $\delta\boldsymbol{I}$ denote the set of items that have changed ratings, one obtains (step 720)

$$\boldsymbol{R}^{t}_{*,\delta\boldsymbol{I}}(t^{-})\delta\boldsymbol{R}(t) + [\boldsymbol{R}^{t}_{*,\delta\boldsymbol{I}}(t^{-})\delta\boldsymbol{R}(t)]^{t} + [\delta\boldsymbol{R}(t)]^{t}\delta\boldsymbol{R}(t)$$

VIII.B. Directly Calculating the Runtime Model

In practice, the matrices $\boldsymbol{R}^{t}\boldsymbol{R}$ and $\boldsymbol{R}\boldsymbol{R}^{t}$ may become extremely large. It is preferable, therefore, to calculate the runtime models iteratively in a manner that allows one to truncate the model at intermediate steps of the calculation. It is also important to note that the approach described here works for both item co-rate models and client co-rate models. This gives the ability to efficiently calculate the top k client neighbors exactly.

VIII.B.1. Item-Based Models

One can consider a partition of all the items for which there exists a client rating. One may denote this partition as $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_k)$ and define (tep 715)

$$\boldsymbol{R}^{(j)}_{u,i} = \begin{cases} \boldsymbol{R}_{u,i} : \text{if } i \in \boldsymbol{B}_j \\ 0 : \text{otherwise} \end{cases}$$

In a manner similar to that used in the parallel dimension model analysis, one may consider *striping* the rating matrix such that (step 715)

$$\boldsymbol{R} = \boldsymbol{R}^{(1)} + \boldsymbol{R}^{(2)} + \ldots + \boldsymbol{R}^{(k)}$$

where $\boldsymbol{R}^{(j)}$ is defined above. The model is given by the following calculation (step 720)

$$\boldsymbol{R}^{t}\boldsymbol{R} = (\sum_{i=1}^{k}(\boldsymbol{R}^{(i)})^{t})(\sum_{j=1}^{k}(\boldsymbol{R}^{(j)}))$$

Collecting terms and reordering the summations yields

$$\boldsymbol{R}^{t}\boldsymbol{R} = (\sum_{i=1}^{k}\sum_{j=1}^{k}(\boldsymbol{R}^{(i)})^{t})(\boldsymbol{R}^{(j)})$$

$$= \sum_{i=1}^{\mathrm{k}}\left\{(\boldsymbol{R}^{(i)})^{t}\boldsymbol{R}^{(i)} + \sum_{j:i<j\leq k}(\boldsymbol{R}^{(i)})^{t}\boldsymbol{R}^{(j)} + (\boldsymbol{R}^{(j)})^{t}\boldsymbol{R}^{(i)}\right\}$$

$$= \sum_{i=1}^{\mathrm{k}}\left\{(\boldsymbol{R}^{(i)})^{t}\boldsymbol{R}^{(i)} + \sum_{j:i<j\leq k}\left\{\boldsymbol{R}^{(i)},\boldsymbol{R}^{(j)}\right\}\right\}$$

where the cross-dimensional terms are defined by

$$\left\{\boldsymbol{R}^{(i)},\boldsymbol{R}^{(j)}\right\} = (\boldsymbol{R}^{(i)})^{t}\boldsymbol{R}^{(j)} + (\boldsymbol{R}^{(j)})^{t}\boldsymbol{R}^{(i)})$$

The runtime model (step 720) may be produced in a variety of ways. One way is to apply the unary operator **Unary**$_k$ as follows

$$\mathbf{Unary}_k(\boldsymbol{R}^{t}\boldsymbol{R}) = \sum_{i=1}^{k}\left\{\mathbf{Unary}_k((\boldsymbol{R}^{(i)})^{t}\boldsymbol{R}^{(i)}) + \sum_{j:i<j\leq k}\mathbf{Unary}_k\left(\left\{\boldsymbol{R}^{(i)},\boldsymbol{R}^{(j)}\right\}\right)\right\}$$

This yields an expression that enables iterative computation of an item-based model.

VIII.B.2. Client-Based Models

In many cases from real data, the matrix of co-rate clients is larger than the matrix of co-rated items and contains more non-zero entries. This results in not being able to store the entire model for co-rated clients, because of memory and storage constraints. In some cases, the same issue applies to the matrix of co-rated items. Hence it is important to be capable of computing the runtime versions of these models. The result of the previous section has a direct consequence to client-based models by replacing $\boldsymbol{R}^t$ for $\boldsymbol{R}$ and noting that $(\boldsymbol{R}^t)^t = \boldsymbol{R}$. Doing so results in the following expression for the client co-rate matrix (step 720)

$$\textbf{Unary}_k(\boldsymbol{R}\,\boldsymbol{R}^t) = \sum_{i=1}^{k}\left\{\textbf{Unary}_k(\boldsymbol{R}^{(i)}(\boldsymbol{R}^{(i)})^t) + \sum_{j:i<j\leq k}\textbf{Unary}_k\left(\left\{(\boldsymbol{R}^{(i)})^t, (\boldsymbol{R}^{(j)})^t\right\}\right)\right\}$$

The above expression and that of the previous section indicate that in spite of memory constraints, the runtime models for co-rates may be iteratively computed. Although doing so makes updates harder, it does not prevent them. In practice, it may be more efficient to re-compute in certain instances rather than computing the incremental update. In either case, the methods employed yield the ability to produce the runtime model directly from the calculation.

IX. THIRD AND FOURTH EMBODIMENTS OF THE PRESENT INVENTION

In a third and fourth embodiment of the present invention, depicted schematically in FIG. 8, a method for providing a recommendation to a client is based upon the ability to perturb a first model so as to generate a second model, or, alternatively based upon the truncation of a first model so as to generate a second model. This corresponds to perturbed model oval 35 as depicted in FIG. 3, or step 820 in FIG. 8. One skilled in the art should appreciate, however, that such a perturbing process may be implemented in a variety of regions in FIG. 3 consistent with the present invention. Accordingly, the methods for perturbing the models include truncation of the model for use in on-line or runtime processing, perturbations that favor a set of items, and functional scaling. All of the methods for perturbing the basic models are derived from either the mathematical structures being used or their internal representation as compressed row matrices.

IX.A. Example of a skewed recommendations

Suppose that one wants to calculate the perturbed recommendation system and an associated marketing campaign. Suppose that widget X is assumed to be in a category Y of widgets. One may wish to perturb the situation so that one is recommending widget X for all the widgets in this category that have profit margins less than that of X's. Let $\boldsymbol{S}_{\mathrm{X}}$ denote the set of widgets that X will replace. Assume that X replaces itself. In order to skew the recommendations, one may construct the following matrix:

$$\boldsymbol{C}_{i,j} = \begin{cases} 1 : \text{if } i = j \notin \boldsymbol{S}_{\mathrm{X}} \\ 1 : \text{if } j = \mathrm{X} \quad i \in \boldsymbol{S}_{\mathrm{X}} \\ 0 : \text{otherwise} \end{cases}$$

Now one needs to compute the perturbed model for on-line recommendations. Two possible matrix products yield skewed models (step 820):

$$\boldsymbol{M^{r'}} = \textbf{Unary}_k\,(\boldsymbol{M}) \bullet \boldsymbol{C}$$

or

$$\boldsymbol{M^{r'}} = \textbf{Unary}_k\,(\boldsymbol{M} \bullet \boldsymbol{C})$$

The marketing campaign for widget X can work directly from ***M*** considering

$$\boldsymbol{R} \bullet (\textbf{Unary}(\boldsymbol{M}_{\mathrm{X},*}))^{\boldsymbol{t}}$$

as a ranking of clients who bought X's neighbors .

Consider further that one has a set of clients ***S*** and one wants to construct a recommendation of the top-*k* items for this set of clients. One possible recommendation is given by (step 835)

$$\textbf{Unary\_Multi\_Vote}_{top\_k}(\textbf{Unary}_q(\ \sum_{u \in U}{}' \boldsymbol{R}_{u,*}), k'\_neighbors) \text{ where } u \in \boldsymbol{S}$$

where the parameter *q* indicates that one is only using the top *q* items purchased by clients from ***S***.

Further still, consider that one has a set of items ***S*** and one wants to construct a recommendation of the top-*k* clients for this set of items. This is a batch process situation. There are two situations to consider. First, a batch process that returns a ranking of clients for each item in the set ***S*** and second, a recommendation of clients for ***S*** entirely.

The first situation is a generalization of the situation presented in the simple scenario, given by

$$\boldsymbol{R} \bullet (\textbf{Unary}(\boldsymbol{M}_{S,*})^{\boldsymbol{t}}$$

and the second is

$$\boldsymbol{R} \bullet (\mathbf{Unary}(\sum_{i \in \boldsymbol{S}}{}' \boldsymbol{M}_{\boldsymbol{S},*}))^{t}$$

One may introduce a *q* parameter by replacing the **Unary** operator with the **Unary**$_q$ operator. Furthermore, either the return vector needs to be sorted or one could only gather clients whose rank in the relative scaling is above a given threshold.

IX.B. Functional scalings

The **Unary**$_k$ function is preferably used for the construction of the on-line models. Recall that, after this operator is applied, the non-zero entries of the resulting vector are equally weighted for use in constructing recommendations. For some data sets, this is fine, but for others, a relative scaling at the individual item neighborhood is relevant to the formation of the final recommendation. In some cases, one may find that instead of using **Unary**$_k$, the use of **Index**$_k$ is more suitable. It has also been useful to scale the individual entries by a weight corresponding to the diagonal terms from $\boldsymbol{R}^t \boldsymbol{R}$. (One skilled in the art should appreciate that the diagonal terms of $\boldsymbol{R}^t \boldsymbol{R}$ are the number of times that an item has been rated.)

Now it is possible to describe a second level of abstraction to making recommendations. Recall that a basic recommendation model used $\boldsymbol{M}^{r(k)} = \mathbf{Unary}_k (\boldsymbol{M})$. If one lets $\alpha \in (0,1]$, then we can scale the entries of $\boldsymbol{M}$ by the diagonal terms to the α root and then use the **Unary**$_k$ operator to construct $\boldsymbol{M}^{r(k)}$. The functional form of this statement is given by (step 820)

$$\mathbf{M}^{r(k)} = \mathbf{Unary}_k\,(\mathbf{M} \bullet ([(\mathbf{D}^2\,\mathbf{M})^{-1}]^{\alpha}))$$

and the basic form of the recommendation remains unchanged as

$$\mathbf{Unary\_Multi\_Vote}_k(\mathbf{S}, k) = \mathbf{TOP}_{k'}\left(\sum_{i \in \mathbf{S}}{}'\left(\mathbf{M}_{i,*}^{r(k)}\right)\right)$$

Next, one may let f be a function defined on the positive real numbers with values in the positive real numbers. Furthermore, one may define the multiple-dimension function given by$\mathbf{F} = (f\,, f\,, \ldots\,, f)$. That is f acts on each coordinate where the dimension is determined by the context. In a preferred embodiment of the present invention, the functional scaling is of the form (step 820)

$$\mathbf{M}^{r(k)} = \mathbf{Unary}_k\,(\mathbf{M} \bullet (\mathbf{F}((\mathbf{D}^2\,\mathbf{M})^{-1})))$$

More generally, let $\mathbf{G} = (g_{i,j})$ and let the recommendations be given by (step 820)

$$\mathbf{M}^{r(k)} = \mathbf{Unary}_k\,(\mathbf{M} \bullet \mathbf{G}(\mathbf{M}))$$

Although this is not the most general functional form, this represents a preferred scaling.

X. FIFTH EMBODIMENT OF THE PRESENT INVENTION

In a fifth embodiment of the present invention, depicted in FIG. 9, a method for providing a recommendation to a client is based upon the construction of a model using cross-set co-occurrences (step 915).

For example, and considering the R-T-R oval 160 in FIG. 3, it may be that the left hand matrix represents a first category of data, and the right handed matrix represents a second category of data. In such a case, the model would then not be $\boldsymbol{R}^t \boldsymbol{R}$, a symmetric matrix, but rather $\boldsymbol{A}^t \boldsymbol{B}$, which in general, is not a symmetric matrix. Such a cross-set model allows for an entirely new basis for recommendations. The first and second matrices for the cross-set co-occurrences, as stated earlier, may be created at any stage of processing, such as sub-space matrices corresponding to particular categories or dimensions.

XI. SIXTH EMBODIMENT OF THE PRESENT INVENTION

Further still, in a sixth embodiment of the present invention, a method for providing a recommendation to a client is based upon the identification of a subset of items through a multiplicity voting scheme, which may be personalized or may be anonymous. This corresponds to, respectively, personal recommendation box 310 of FIG. 3 or anonymous recommendation box 320.

XI.A. Making Anonymous Recommendations

One of the purposes of the calculations is to construct a recommendation model with a desired property. A suitable set of models is constructed in a sequence of off-line processing stages. This separation of processing minimizes the runtime evaluation as described by the functions $\mathbf{Unary\_Multi\_Vote}_{k'}(*, k)$ and $\mathbf{Multi\_Vote}_{k'}(*, k)$

Of course, any additional processing of utility may be further applied. However, one focus of the present invention is to minimize the need for such runtime processing by suitably constructing the models such that minimal runtime processing is required. Note that the evaluation of **Unary_Multi_Vote** or **Multi_Vote** involves adding of vectors from a matrix that represents the on-line model.

XI.B. Making Personalized Recommendations

The making of a personalized recommendation is an application of anonymous recommendation, in which the system first constructs a list of items that are personalized and uses this list in the anonymous recommendation strategies.

XII. GENERALIZED ASPECTS OF THE PRESENT INVENTION

The remainder of the discussion herein will focus primarily on generalizations surrounding aspects of the invention as described in the preferred embodiment. In turn, these generalizations encompass the topics of filtering, high-order state spaces, and data properties. The topic of sparse matrix calculations is also briefly discussed.

XII.A. Generalized Filtering: Item Injection/Rejection

The issue of filtering may be done in many ways—one example is given in the usage scenario. In that case, a matrix was produced that had an effect on the manner in which the recommendations would be created. This type of an effect can be described as a scaling factor. This will be described formally in the next section.

XII.A.1. Scaling Factors

As an example, the type of scaling factor that was used in the simple usage scenario is described here. Recall that there was a rule to replace widget X for all widgets in $\boldsymbol{S}_X$. This rule induces a $\{0,1\}$-valued function of the set $\{(i, j) \mid i, j \in I\}$. That is, a value of one if the rule holds between *i* and *j*, and a value of zero otherwise. Such functions directly give rise to matrices that may be used as scaling factors.

$$W = W_{i,j} = \begin{cases} 1 : rule(\, i,j) = 1 \\ 0 : \text{otherwise} \end{cases}$$

In fact, a matrix that encodes a usual scaling of items to items may be used as a scaling factor

XII.A.2. Item Rejection/Injection

As a direct consequence of producing a model, there are multiple places to perform item rejection or item injection. As described, recommendations can be made from a class of models. Entries in these models may be removed, rejected, or have their co-rate values modified in order to meet some rule-based policy to be enforced. As such, the need to perform such roles becomes less stringent at the runtime evaluation of the recommendations. Doing such modifications at the model level can be performed in either a reversible or irreversible manner. This is the choice of recommendation policy that the recommendation engine will enforce. Note that any such preprocessing of data does not rule out the possibility of runtime evaluation of rules that enforce the relevant portions of a runtime recommendation policy.

XII.B. Generalized High Order State Spaces

All of the previous examples discussed here dealt with, at most, integer-valued matrices. However, this does not have to be the case. There are three areas in which a more general model will be of immediate utility and are consistent with the present invention. These areas are the following: (i) “Not For Me:” a scenario in which clients of a web site indicate that an item should not be recommended to them, and amounts to incorporating negative feedback into the model based prediction algorithms; (ii) “Temporal Data:” where, in the descriptions of calculating the model, there has been no use of time; and (iii) “Windowing:” a scenario in which the model reflects data collected only after a certain data and one wants to maintain the model's accuracy as a running model.

All three of these scenarios have a common structure that is described below. One important aspect of this is the idea of pointing to a value of a entry of a matrix. That is, one can separate the reference of the (i, j)-entry of a matrix from the value. As a result, the value need not be integer-valued. Thus, this approach can be easily extended to support vector-valued matrices or some other useful structure. Described below are extensions in which the use of vector-valued matrices is both natural to consider and of utility.

XII.B.1. “Not For Me”

The notion of “not for me” is basically a three-state model for the values of a client's preference between any two items. In the initial formulation there were two states, *rated both* and *no information*. The reason the second state is referred to as *no information* is if $a \bullet b = 0$, then $\{ a = 0, b = 1\}$, $\{ a = 1, b = 0\}$, and $\{ a = 0, b = 0\}$ are indistinguishable states of a and b. If it is desirable to distinguish between these states (gaining information) then one must extend the notion of $a \bullet b$ to take values in a larger state space. In the case of introducing “not for me” this is exactly what one wants to do.

One may consider a state space of zero-one-valued triples, $\{ 0, 1\}^3$, whose entries some to zero or one. There are only possibilities, $\Omega = \{(0, 0, 0); (1, 0, 0); (0, 1, 0); (0, 0, 1)\}$. We will allow our ratings to take values in $\Omega_{admissible} = \{(0, 0, 0); (0, 0, 1); (0, 1, 0)\}$. The value (1, 0, 0) will be reserved to indicate disagreement. A dictionary for these ratings values may be:

(0,0,0) indicates no information for the client. Of course, using sparse matrices, we do not store these values;

(0,1,0) indicates that the client rated the item favorably; and

(0,0,1) indicates that the client rated the item "not for me."

The overload multiplication as defined by Table C:

Table C

	(0, 0, 0)	(0, 1, 0)	(0, 0, 1)	(1, 0, 0)
(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	-
(0, 1, 0)	(0, 0, 0)	(0, 1, 0)	(1, 0, 0)	-
(0, 0, 1)	(0, 0, 0)	(1, 0, 0)	(0, 0, 1)	-
(1, 0, 0)	-	-	-	-

where the "-" indicates the non-admissibility of the ratings being "multiplied." Summation of any two elements from Ω is performed coordinate wise. The following could equally have been defined over complex-valued matrices, but in practice the above description would be more practical. If $\boldsymbol{R}$ is a ratings matrix taking values in $\Omega_{admissible}$, the set of admissible states then

$$\boldsymbol{R}^t * \boldsymbol{R} = (\boldsymbol{R}^t * \boldsymbol{R})_{i,j} = \sum_u \boldsymbol{R}^t_{i,u} * \boldsymbol{R}_{u,j}$$

and defines our state matrix of co-rates.

Next is the issue of making recommendations. Previously, the approach was to take the top co-rated entries of a row vector. If one wishes to use only one coordinate from the state vector this scheme will still work, but this new situation is far more extensible than that simple situation. A possible scenario for distinguishing between state vectors is given as follows.

Let $\Psi = (\psi_1, \ldots, \psi_n)$ denote the state vector from $\boldsymbol{R}^t * \boldsymbol{R}$, in which Ω is the state space for ratings. Furthermore, let f be an non-negative integer-valued function on Ω. For example, we may want to define cut-off thresholds. In order to properly define such examples, consider the Heavyside function defined on the real values.

$$\boldsymbol{H}(s - t) = \begin{cases} 1 : \text{if } s - t \geq 0 \\ 0 : \text{otherwise} \end{cases}$$

Now one can discuss functions such as

$$f^{k,k'}(\psi) = \psi_{(0,1,0)} \boldsymbol{H}(k - \psi_{(0,1,0)}) \boldsymbol{H}(k' - \psi_{(1,0,0)})$$

This function evaluates to the favorable co-rate value, $(\psi_{(0,1,0)})$, as long as the value the "not for me" and "disagreement" value are not above the cut-off values of k, k' respectively.

The above function is exemplary only, however, and in no way limits the present invention.

XII.B.2. Temporal Data

Further still, one can consider the problem of making the state time dependent. Recall that in Section VIII.A the incremental update for any change in the ratings was computed.

Suppose that $\boldsymbol{S}$ is defined to be real-valued pairs. The first entry stores the increment, and the second entry stores the timestamp indicating when the update occurred. Now rather than have a matrix that only indicates the co-rate value in its entries, one has a co-rate matrix whose entries are values in sequences in $\boldsymbol{S}^{\infty}$ whose coordinates are all zero for sufficiently large indices, and which will encode that there have been only a finite number of updates in any time interval. The first entry will store the current value of the co-rate and the remaining entries of the sequence keep track of the incremental updates. In this manner the temporal nature of the co-rating matrix has been maintained.

XII.B.3. Windowing

Further still, in practice, the description of sequencing the incremental updates will become storage intensive and uninteresting. As a result of maintaining timestamps, it is possible to incrementally remove co-rate contributions that are too old. This may be performed at the time of an incremental update by checking whether there are any previous updates that are *out of date*. This is useful for item identifiers in the model that have been reused to represent similar items. One does not want to remove all the rating for the item, because then the start up problem exists. In this phasing of valid co-rates, we eventually have co-rates that represent the proper item in the real world (as opposed to this matrix representation of the world).

XII.C. Generalized Data properties

Throughout the discussion presented here, *unary* data, as defined herein, has been the predominate data form. However, and in relation to the step of preprocessing data in a preferred embodiment, “Interest Data” and “Likert Data” are discussed briefly.

XII.C.1. Interest Data

The fundamental fact that makes $\boldsymbol{R}^t\boldsymbol{R}$ and $\boldsymbol{RR}^t$ calculations possible is that if $a, b, \in \{0,1\}$ then $ab = 1 \Leftrightarrow a = 1$ and $b = 1$. Note that $ab = min(a, b)$. If one considers the value “1” to mean *interested in the item* and the value “0” to mean *no information*, then the co-rate contribution of two entries makes sense as their minimum. Consider overloading the matrix multiplication operation (denoted with *):

$$\boldsymbol{A} * \boldsymbol{B} = (\boldsymbol{A} * \boldsymbol{B})_{i,j} = \sum_{u} min(\boldsymbol{A}_{i,u}, \boldsymbol{B}_{u,j})$$

All of the methods described are immediately extended to situations in which an overloaded multiplication operation is used. Note that if the rating data is in fact unary, this operation reduces to the unary algorithms describe elsewhere in this document. This fact has a direct consequence for the parallel dimensions algorithms. Suppose that one has three dimensions, one dimension of which is unary and two dimensions of which are interest. Recall that the cross dimension models were defined by

$$M(i,j)^{r(k)} = \textbf{Unary}_{k}((R^{(i)})^{t}R^{(j)})$$

The cross dimension models may now be extended to

$$M(i,j)^{r(k)} = \textbf{Unary}_{k}((R^{(i)})^{t} * R^{(j)})$$

Suppose that the *j* dimension is an interest dimension and *i* dimension is unary, then

$$\textbf{Unary}_{k}((R^{(i)})^{t} * R^{(j)}) = \textbf{Unary}_{k}((R^{(i)})^{t}\,\textbf{Unary}(R^{(j)}))$$

This formula shows that in cross-dimension models involving a unary dimension, the model will reduce to the cross dimension model where the interest dimension in question has had a unary operator applied to make the data unary. This results in being able to deal seamlessly with multiple dimensions, mixed between interest and unary ratings.

XII.C.2. Likert-Binary Data

The approach to dealing with Likert data has been to make unary ratings data from the Likert data by using a binary cut-off. That is, values above a predefined threshold become one and otherwise zero. This threshold may be done on a client-to-client basis.

XIII. EXAMPLES OF SPARSE MATRIX CALCULATIONS

One aspect of the present invention has been the focus on the mathematical formulation for calculating co-occurrence in ratings data. The formulation disclosed used standard mathematical techniques in order to derive formulas that have utility for making fast, accurate recommendations. Since, most often, the data from which the recommender system will need to compute is extremely sparse in its matrix representation, it is further useful to turn attention towards some numerical analysis aspects. An example of a numerical analysis kit which may be used with the present invention is **SPARSEKIT2** (available from the University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN, <ftp://ftp.cs.umn.edu/dept/sparse/>).

XIII.A. Compressed Sparse Row Format (CSR)

The matrices that have been discussed are such that the non-zero entries on a row represent a client's ratings. As such it is not surprising that one desires a representation for the computation that favors a viewpoint that is efficient for accessing the non-zero entries of a row. Such accesses are not the only operations that are required. Matrix multiplication is a fundamental operation and hence one will need this operation to be efficient. The compressed sparse row format, described below, is quite suitable for computation of the calculations discuss earlier. For a more thorough description of this and other formats the reader is directed to the documentation of the **SPARSEKIT2** package.

The data structure used to represent a compressed sparse row formatted matrix consists of three arrays: (i) an array containing the non-zero entries of the matrix; (ii) an array containing the column positions of these non-zero entries; and (iii) an array containing pointers into the previous arrays corresponding to the beginning of each row of the matrix.

As an example, suppose that the matrix R in its standard representation is given by

$$\boldsymbol{R} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Consider that the following arrays are zero based. FIG. 10 shows three arrays that represent the matrix ***R*** in compressed-row format. Note that the last value of the array of row pointers is a reference to where the next row would begin if there were a next row. In essence, it encodes how many non-zero entries exist on the last row.

XIV. CONCLUSION

Methods and apparatus consistent with the present invention can be used to provide rapid, accurate, preference recommendations to a client. The foregoing description of an implementation of the invention has been presented for purposes of illustration and description. It is not exhaustive and does not limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practicing the invention. For example, although the runtime recommendation system and off-line recommendation system were depicted as separated by a network (wireless or otherwise), such a depiction was exemplary only. One skilled in the art should appreciate that runtime recommendation system and off-line recommendation system may form different processing and memory portions of the same data processing device. Accordingly, the invention is not limited to the above described embodiments, but instead is defined by the appended claims in light of their full scope of equivalents.